Grid Resource Assessment from Statistical Indexes

Nicolas Dubé and Marc Parizeau

Electrical and Computer Engineering Department
Université Laval, Québec, Canada, G1K 7P4

Abstract: This paper presents an innovative Grid Market Exchange pricing model that embodies advance reservation, component integration, neighborhood valuation and quality of service feedback. In the context of an upcoming *Grid Economy*, it is able to appraise job requirements and node components from historical market information. These resource indexes constitute valuable estimation tools for compute time consumers and providers.

Key-Words: grid, resource, node, job, pricing, valuation, assessment, market, economy, requirement, component

1 Introduction

With the advent of high-bandwidth backbones in the late 90's, computer scientists started dreaming about and designing a worldwide federation of computing resources and sensors: *The Grid* [6, 7, 8]. In this grid, every single compute node, scientific sensor or data repository would be virtualized and accessible to anybody, anywhere, anytime. It is taking the Internet to the era of wide-scale distributed computing, transparent remote execution and *computing on demand*.

In the last decade, large compute grids have been made possible thanks to substantial government infrastructure investments, just as electrical networks came to light in the 40's and 50's. Funding was provided to visionary researchers in the field to design and implement the emerging grid. But building this kind of architecture involved high risk levels and the resulting product was far too unstable to be accounted for rigorously. Coupled with the grid community's idealistic views, this risk explains the actual free and undervalued nature of grid CPU power.

Out of its implementation transient state, the grid infrastructure is about to provide a stable and definable quality of service. CPU power will then be considered as a valuable good that can be assessed over time. *Grid Economy* [3] is an emerging research field trying to define computing resources exchange policies and systems to put them in place. While a market-driven CPU economy involves building meta-schedulers to provide dynamic accounting, auditing and advance reservation of resources, and before even thinking of trading CPU stocks, a framework composed of publishers, brokers, banks and monitors must be implemented.

The GridBus [4] project is developed at the Grid Computing and Distributed Systems (GRIDS) Laboratory of the University of Melbourne, Australia (project leader and laboratory director, Rajkumar Buyya, is a leading Grid Economy scientist [1]). GridBus is an all-inclusive system that regroups the many components (GSB, GMB, GridBank, G-Monitor, GridSim) targeting the same objective: building the grid economy infrastructure. Other projects like OCEAN [9], Nimrod-G [2], and Libra [10] are also dealing with various aspects of the grid market exchange.

The primary trend supporting a market-driven grid, as most *Grid Economists* would argue, refers to a fundamental incentive toward computing power exchange. Currently, compute time-slot exchange is based mostly on a system of good-will. While this “socialist” perspective can hold for demonstration purposes, on a day-to-day basis, computing power producers and consumers need a *user-centric* system implementing schedulers and accounting with a definable quality of service.

Many auction models can apply to this transactional model, either English, Dutch or double auction, but all require a deterministic assessment of the true value for a given time-slot on a specific node. One major problem is that computing power producers (supercomputer owners, LAN managers, etc.) have no clear way of estimating how much their CPU cycles are really worth. Consumers, mostly scientists from other fields, are even further from this assessment.

To be a stable and maintainable wide-scale system, the Grid Market Exchange must be self-regulating and reduce management overheads to a minimum. While conventional markets trade stocks based on a current value, the Grid Market Exchange will need to support *advanced reservations*, in a manner similar to stock options. But it would then be impossible for anybody to appraise every single time-slot on a cluster for the months to come, a process far too speculative and time-consuming. Whether the exchange unit is based on a real monetary mapping or on some form of fictive currency, we must have automated tools based on statistical data to estimate the cost of resources.

2 Objectives

In this paper, we advocate a Grid Market Exchange broker implementing a statistical pricing algorithm. To be deployable on a wide-scale grid, this broker will have to support the following:

2.1 Advance Reservation

On a worldwide grid supporting advance reservations, every single future time-slot for a specific node would become an exchangeable unit on a compute time market. Auction models involve individual time-slot assessment; thus for a group of 512 processors and 1 hour time-slots, this represents 4 485 120 “cpu stocks” to appraise per year. Even on a homogeneous cluster, disregarding configuration aspects, the owner must still assess 8 760 values (24 hours x 365 days). Since offer and demand will vary dynamically over time, this process would have to be re-evaluated regularly, probably on a daily basis, a daunting task. This is the fundamental motivation for some form of automation in the pricing algorithm.

2.2 Component Integration

While most actual grid applications specify relatively few configuration requirements (processor time, available memory and disk space), future applications might necessitate additional constraints like memory bandwidth, cache size, etc. The pricing model must therefore be adaptive, to enable the seamless integration of new trends in application requirements. Furthermore, the relative significance of components will undoubtedly vary over time, and node pricing will have to be adjusted accordingly. Such a model would therefore be able to make the transition to any future breakthrough in HPC technology.

2.3 Neighboring Nodes

While embarrassingly parallel apps can compute on distant nodes almost without consideration for the interconnect, many MPI programs need tightly coupled nodes. Therefore, a compute node linked to similar nodes with Myrinet or InfiniBand will undoubtedly present more value than a comparable broadband node. This reality must be reflected in the pricing model. It must also consider dynamically evolving interconnect aspects of throughput and latency.

2.4 Quality of Service

This aspect is certainly the cornerstone of the Grid Economy. Quality of service will have to be advertised by producers regarding every node component. In fact, compute node owners will sell a given level of service for their nodes. Consumers, depending on how much they will be willing to pay, will agree to these terms and anticipate an appropriate fulfillment. Feedback mechanisms will have to be put in place to evaluate advertised QoS and apply corrections if needed.

3 Pricing Model

While many have worked on market-driven compute time brokers and schedulers [2, 5, 10, 11], few pricing models exist as it is considered that human interaction would determine bid and ask prices. Recent work by Yeo and Buyya [12] propose a pricing model for utility-driven management and allocation of resources on a cluster. Their model implements a fair accounting scheduler and broker where users actually “pay” for their effective resource needs using the following equation:

$$P_j^n = \left(\alpha + \beta U_j^n\right) P_{\text{base}}^n, \tag{1}$$

where the unit price P_j^n for a given resource on a cluster node n for a job j is computed from a base price P_{base}^n (set by the owner) for that resource, and a utilization factor U_j^n determined at run-time:

$$U_j^n = \frac{R_{\text{max}}^n}{R_{\text{free}}^{jn}}, \tag{2}$$

where

$$R_{\text{free}}^{jn} = R_{\text{max}}^n - \sum_{i \neq j} R_i^n, \tag{3}$$

with R_{max}^n and R_i^n being the maximum amount of this resource for node n, and the amount consumed by another job i running on the same node, respectively.

While this pricing function generates a higher price under high demand for the provider, there is no assurance for the consumer that the provider won't load the resources at run-time (deliberately or not), reducing R^{jn}_{free} therefore raising U^n_j.

This model also implies dynamically measuring job execution-time, memory and storage usage ratios. The α and β factors provide flexibility over the weighting of static and dynamic components in the overall pricing. They have proven higher cluster profitability with enhanced waiting time and response time.

Although this model has given good results on an isolated cluster, we are willing to define a pricing model in a wider perspective. The Grid Economy will be based on a market exchange matching Task Level Agreements (TLAs) with Service Level Agreements (RLAs) before the job execution. Therefore, the initial deal cannot be made over effective resource utilization for the job although this information will be very useful a posteriori.

3.1 Pricing as a Time Function

We advocated earlier that compute nodes are far too numerous to be assessed on a one by one basis. Moreover, for an advance reservation CPU market-exchange, we must consider the temporal aspect of compute power. Therefore, the pricing function $\mathcal{P}$ for a given distributed application would have to take this general form:

$$\mathcal{P}(t_0, t_1, t_2) = \int_{t_1}^{t_2} P(t)dt, \tag{4}$$

where we must integrate over time the cost at t_0 of each time-slot needed by the job to complete from $t_1 = t_0 + \Delta t$ to $t_2 = t_1 + \lambda t$. The total execution price for the application therefore depends on the deal time t_0, the delay between deal time and launch time Δt and total execution time λt. For a set of jobs J running on a set of nodes N, we can define a pricing function $P(t)$ for a time-slot t as:

$$P(t) = \sum_{n \in N} P_n(t) = \sum_{j \in J} P_j(t), \tag{5}$$

where $P_n(t)$ is the price of node n and $P_j(t)$ is the price for job j. For every j running on n, when a deal is concluded, $P_j(t) = P_n(t)$. Therefore, equation 5 is true for any one-to-one mapping of J and N.

3.2 Fundamental vs Speculative Inputs

For every compute node, we argue that autonomic valuation algorithms should be based on a fundamental trend calculated on original and operational costs, on the one hand, and a market-driven speculative component, on the other hand. Intuitively, as shown in Figure 1 (solid line), we can think of computing power starting at a base price level in a distant future, growing incrementally approaching present time, and probably dropping drastically sometime before the expiry of the time-slot. This behavior is somewhat comparable to airplane ticket prices or other commodity markets. This reflects the "blue-chip" aspect of long term bidding, the speculative increment, and the fact that remaining cpu cycles, if not sold, will be lost.

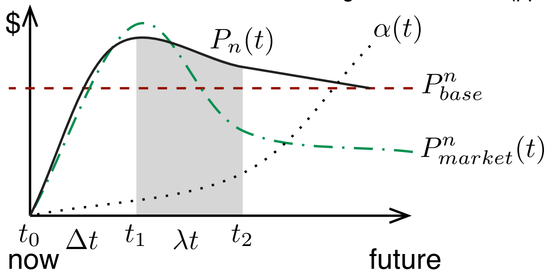

Figure 1: *Grid Pricing Graph*

We therefore propose a pricing function presenting these two components: basic (fundamental) and market (speculative):

$$P_{n/j}(t) = \alpha(t)P^{n/j}_{\text{base}} + (1 - \alpha(t))P^{n/j}_{\text{market}}(t), \tag{6}$$

where the price for a node n or a job j at time t is determined by a static basic price $P^{n/j}_{base}$ and a dynamically evolving speculative index $P^{n/j}_{market}$ for that specific node configuration or set of job requirements. P^n_{base} is assessed by the node owner upon original and maintenance costs. Likewise, the consumer can determine a basic price, P^j_{base}, he is willing to pay for compute time-slots in the long run. Recall that when $P_j(t) = P_n(t)$, we have a deal. The α parameter serves to tweak the relative significance of fundamental versus speculative inputs. It might in fact be a function of t and could decrease linearly or exponentially. This parameter shall be determined by the resource owner and, similarly, by the compute time consumer.

3.3 Component Aggregation Pricing

It is also very important to consider node component relationships. Quality of service must not be calculated solely on resource availability, it must also consider efficiency. Therefore, many more parameters must be evaluated at run-time and considered in the pricing function.

They can be static characteristics (memory throughput, cache size, hard drive interconnect, hard drive throughput, etc.) or run-time benchmarks (Linpack, STREAM, etc.). For example, since memory bandwidth is becoming a major roadblock for many HPC applications, a truly dynamic and adaptive pricing model would probably incorporate it. Therefore, the individual time-slot price P^n_{market} for a node n can be evaluated by summing the market indexes of its components:

$$P^n_{\text{market}}(t) = \sum_{c \in C} I^c_{\text{comp}}(t), \tag{7}$$

where C is the set of node components and I^c_{Comp} is the market index of component c. The market value of node n is therefore less speculative in some manner because the node's pricing comes from the sum of its speculative parts, representing larger scale market trends thanks to an enhanced granularity. As we will see later, component and requirement indexes might not be equal, we thus define $P^j_{\text{market}}(t)$ for a job j as:

$$P^j_{\text{market}}(t) = \sum_{r \in R} I^r_{\text{req}}(t), \tag{8}$$

where R is the set of job requirements and $I^r_{\text{req}}(t)$ is the market index of requirement r.

3.4 Neighborhood Valuation

Another important aspect not considered in current pricing models is the proximity of similar compute nodes and storage hardware. For an embarrassingly-parallel problem, this presents very little impact. For tightly-coupled ones, it is a whole different story. Every scientist in the HPC/Grid community knows that data-driven applications or tightly-coupled problems are more expensive to run than embarrassingly-parallel ones. They require high-capacity storage and massive network throughput, respectively. A more realistic pricing model will have to consider resources interconnect.

Therefore, we introduce the concept of neighborhood resources that must be counted like other components or requirements in equation 7 and 8. Consumers and providers would therefore be able to specify the number of nodes, interconnect and latency specifications, in their TLAs and SLAs, respectively.

3.5 Quality of Service Feedback

When executing a job, quality of service validation mechanisms will have to be applied to make sure the consumer gets what he paid for. This is where the utilization factor comes into play. If a user paid for 90% of a compute node CPU power, the producer must satisfy the requirement. In this regard, it would be interesting to modify Equation 7 to incorporate an advertised resource utilization factor δ_c, that could be determined automatically upon prior job runs or manually by the provider.

$$P^n_{\text{market}}(t) = \xi_n \sum_{c \in C} \delta_c I^c_{\text{comp}}(t). \tag{9}$$

Finally, user satisfaction and vendor commitment are clearly aspects with an intrinsic value in every economy. QoS information feedback on the computational market exchange is necessary because trustable grid nodes are clearly of higher value than unstable ones. A final factor ξ_n is thus added as an averaged trust index computed from the node QoS history.

4 Resource indexes

The need for some form of automated pricing algorithm comes from the impossibility for any *grid broker* to appraise in real-time every future time-slot for every compute node. This comes from the far too numerous time-slots but also from the combinatorial explosion of node configurations. Therefore, if we want to appraise any given node, we argue that we must first decompose previous deals into discrete resource indexes from which we can concatenate various individual node pricing.

4.1 Matching requirements and components

We start with a set of resources $\mathbf{\Phi}$ from which job requirements $R \subset \mathbf{\Phi}$ and node components $C \subset \mathbf{\Phi}$ can be expressed. These resources can be decomposed in subsets of discrete resource types like memory size, disk space, CPU architecture and frequency, memory bandwidth, cache memory, benchmark results, software stack components, etc. It is important to note that $\mathbf{\Phi}$ is not a fixed set, but can evolve dynamically over time. For example, we could consider the following:

$$\mathbf{\Phi} = \Phi_{\text{mem}} \cup \Phi_{\text{disk}} \cup \Phi_{\text{cpu}} \cup \Phi_{\text{freq}} \cup \Phi_{\text{nodes}} \cup \Phi_{\text{misc}}$$

with:

$$\begin{aligned}
\Phi_{\text{mem}} &= \{128, 256, 512, 1024\} && \text{(MB)}\\
\Phi_{\text{disk}} &= \{1, 2, 4, 8\} && \text{(GB)}\\
\Phi_{\text{freq}} &= \{1.0, 2.0, 3.0\} && \text{(GHz)}\\
\Phi_{\text{cpu}} &= \{\text{x86}, \text{ppc}, \text{sparc}\} &&\\
\Phi_{\text{nodes}} &= \{7@10, 31@1\} && \text{(n@Gbit)}\\
\Phi_{\text{misc}} &= \{\text{vpu}, \text{gcc4}, \text{mpi}, \text{globus}\} &&
\end{aligned}$$

Therefore, a user could define the following requirements for an individual job j with only a small subset of $\mathbf{\Phi}$:

$$R_j = \{512_{\text{mem}}, 1_{\text{disk}}, \text{ppc}_{\text{cpu}}, 7@10_{\text{nodes}}\} \tag{10}$$

This would denote a job compiled for the PPC architecture, necessitating 512MB of RAM and 1GB of disk storage and requiring 7 similar nodes linked with a 10 Gbit interconnect. Note that this user does not specify any requirement regarding performance, this could be part of the decision making strategy to obtain the lowest price point for total execution. We will return to this later. Similarly, the owner of node n defines its components:

$$\begin{aligned} C_n = \{&256_{\text{mem}}, 512_{\text{mem}}, 1024_{\text{mem}}, \\ &1_{\text{disk}}, 2_{\text{disk}}, 4_{\text{disk}}, 8_{\text{disk}}, \\ &1_{\text{freq}}, 2_{\text{freq}}, \text{ppc}_{\text{cpu}}, 7@10_{\text{nodes}}\} \end{aligned} \tag{11}$$

This node would therefore accept jobs requiring 256 MB, 512MB or 1GB of RAM (but not 128 MB) on a PowerPC architecture from 1GHz to 2GHz, necessitating between 1GB and 8GB of storage. It can also be sold with 7 similar 10 gigabit nodes. Hence, the set A_j of admissible nodes for job j (defined by requirements R) is given by:

$$A_j = \{n \in N \mid R_j \cap C_n = R_j\} \tag{12}$$

4.2 Market-driven pricing feedback

At the creation of a grid brokering infrastructure, all pricing information will be based upon the various P^n_{base} set by the providers and the P^j_{base} of consumers. As soon as the first deal is finalized, global market exchange information will be used to compute resource indexes and consequently market-driven pricing. With statistics on requirements and components in a set of previously concluded deals $\mathcal{D}$, we can maintain a database containing time relevant counts for every requirement $\eta^{\phi}_{\text{req}}(t)$ and component $\eta^{\phi}_{\text{comp}}(t)$. Using historical statistics, we can therefore calculate resource market ratios:

$$\rho_{\phi}(t) = \frac{\eta^{\phi}_{\text{req}}(t)}{\eta^{\phi}_{\text{comp}}(t)}, \quad \forall \phi \in \mathbf{\Phi} \tag{13}$$

Table 1 shows sample statistics from requirements of 64 executed jobs and components offered on 64 heterogeneous nodes. All 64 jobs required some amount of memory and a specific kind of processor, 42 of them required some disk space and 36 a specific CPU clock. We can see that 64 nodes had at least 512MB of memory but 4 of them were not willing to compute smaller jobs requiring 128MB of RAM. As CPU architecture is a binary input, processor type statistics are mutually exclusive.

Thus, we can compute component and requirement pricing indexes by retro-propagating every single deal price $\Pi_{\$}$ proportionally:

$$I^c_{\text{comp}}(t) = \rho_c(t) \;/\; \sum_{\phi \in C} \rho_{\phi}(t) * \Pi_{\$} \tag{14}$$

$$I^r_{\text{req}}(t) = \rho_r(t) \;/\; \sum_{\phi \in R} \rho_{\phi}(t) * \Pi_{\$} \tag{15}$$

Taking the previous example (eq. 10-11) and considering the deal had been closed at 1000 grid units (gu):

$$I^{512_{\text{mem}}}_{\text{comp}}(t) = \frac{0.44}{3.60} * 1000gu = 121gu \tag{16}$$

$$I^{512_{\text{mem}}}_{\text{req}}(t) = \frac{0.44}{2.25} * 1000gu = 196gu \tag{17}$$

Resource pricing indexes are therefore calculated dynamically over time. From a system point of view, the real-time value for an index could be simply its last deal value or an averaged combination of surrounding deal values in time.

The motivation for a double index scheme comes from the fact that the brokering system will not match job requirements on limited node components. In fact, in almost every deal, the consumer will pay in some way for some components not listed in the requirements, still presenting a positive market ratio (eq. 13). Although we must interpolate every component index when closing a deal on a given node, the requirement indexes must be higher to compensate for unrequested, but still unusable by others, components.

In a steady-state system, the consumer will be informed in real-time of the lowest, average and highest priced (ask) available node matching its requirements. Helped by the market-driven index $P^j_{\text{market}}(t)$, he will be able to make a better buying decision. Similarly, the provider will be presented the lowest, average and highest priced job (bid) able to run on its node, using $P^n_{\text{market}}(t)$, he will thus be able to steer its selling price. In both cases, the decision making strategy could be automated to match a given percentage of resource indexes depending on the execution timeframe, for example.

Resource	η^{ϕ}_{req}	$\eta^{\phi}_{\text{comp}}$	ρ_{ϕ}	I^{ϕ}_{req}	I^{ϕ}_{comp}
128_{mem}	10	60	0.17		
256_{mem}	18	64	0.28		78 gu
512_{mem}	28	64	0.44	196 gu	121 gu
1024_{mem}	8	32	0.25		69 gu
1_{disk}	20	64	0.31	138 gu	87 gu
2_{disk}	15	64	0.23		64 gu
4_{disk}	6	64	0.09		25 gu
8_{disk}	1	41	0.02		6 gu
1.0_{freq}	12	64	0.19		53 gu
2.0_{freq}	16	56	0.29		80 gu
3.0_{freq}	8	0	0.17		
$x86_{\text{cpu}}$	48	48	1.00		
ppc_{cpu}	8	8	1.00	444 gu	278 gu
$sparc_{\text{cpu}}$	8	8	1.00		
$7@10_{\text{nodes}}$	8	16	0.50	222 gu	139 gu
$31@1_{\text{nodes}}$	0	48	0.00		
	64	64		1000 gu	1000 gu

Table 1: Example of resource statistics.

5 Conclusion

This paper presented an innovative grid market exchange pricing model that embodies advance reservation, component integration, neighborhood valuation and quality of service feedback. Most introduced ideas are in fact part of a work in progress and an exhaustive simulation engine to validate the model on a larger scale is being developed. Many other calculation approaches for the resource indexes can be envisioned and will be evaluated in future work. A prototype infrastructure implementing this market-driven brokering model will be deployed through various testbeds in North-America.

On a wider perspective, further research will be conducted on the similarities between a CPU exchange market and other commodity economies like electricity, airplane tickets, coffee, bananas, etc. This could bring in some very interesting insight on how an upcoming wide scale CPU time-slot market would behave. In the long term, a Grid Market Exchange could outline the real needs of users, needs that hardware vendors will be able to use to build better computers, instead of relying on actual flawed benchmarks.

The *Grid Economy* will bring a plethora of new economic and computer science phenomenons to analyse. While it might favor the Grid democratization, consumers and providers will be mostly newcomers to this market exchange. We must therefore envision innovative foundations of trust in this economy; assessing resources statistically could be one of them.

References

[1] Rajkumar Buyya. *Economic-based Distributed Resource Management and Scheduling for Grid Computing*. PhD thesis, Monash University, Melbourne, Australia, 2002.

[2] Rajkumar Buyya, David Abramson, and Jonathan Giddy. Nimrod-G resource broker for service-oriented Grid computing. *IEEE Distributed Systems Online*, 2(7), 2001.

[3] Rajkumar Buyya, David Abramson, and Srikumar Venugopal. The grid economy. *Special Issue on Grid Computing, Proceedings of the IEEE*, 93(3):698–714, March 2005.

[4] Rajkumar Buyya and Srikumar Venugopal. The gridbus toolkit for service oriented grid and utility computing an overview and status report. In *1st IEEE Int. Workshop Grid Economics and Business Models (GECON 2004)*, 2004. Comment: 11 pages, 3 figures, 3 tables.

[5] Brent N. Chun and David E. Culler. Market-based proportional resource sharing for clusters, March 1999.

[6] Ian Foster, C. Kesselman, J. Nick, S. Tuecke, and S. Fitzgerald. The physiology of the grid: An open grid services architecture for distributed systems integration. *Open Grid Service Infrastructure WG, Global Grid Forum*, page 31, 2002.

[7] Ian Foster and Carl Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, San Francisco, CA, USA, second edition, 2004.

[8] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organization. *The International Journal of High Performance Computing Applications*, 15(3):200–222, Fall 2001.

[9] Pradeep Padala, Cyrus Harrison, Nicholas Pelfort, Erwin Jansen, Michael P. Frank, and Chaitanya Chokkareddy. Ocean: The open computation exchange and arbitration network, a market approach to meta computing. In *Proceedings of the ISPDC'03 (International Symposium on Parallel and Distributed Computing)*, 2003.

[10] Jahanzeb Sherwani, Nosheen Ali, Nausheen Lotia, Zahra Hayat, and Rajkumar Buyya. Libra: a computational economy-based job scheduling system for clusters. *Software, Practice and Experience*, 34(6):573–590, May 2004.

[11] I. E. Sutherland. A futures market in computer time. *Communications of the ACM*, 11(6):449–451, 1968.

[12] Chee Shin Yeo and Rajkumar Buyya. Pricing for utility-driven resource management and allocation in clusters. In *Proceedings of the 12th International Conference on Advanced Computing and Communication*, Ahmedabad, India, December 2004.